Foucault Pendulum Electronics Kit.

D03_Functionality of the GUI_Pendulum program

www.foucaultpendulum.nl

Document version	2026-06-08
Related Documents	D02_Options Parameters

In brief:
This document describes the functionality of the GUI_Pendulum program.

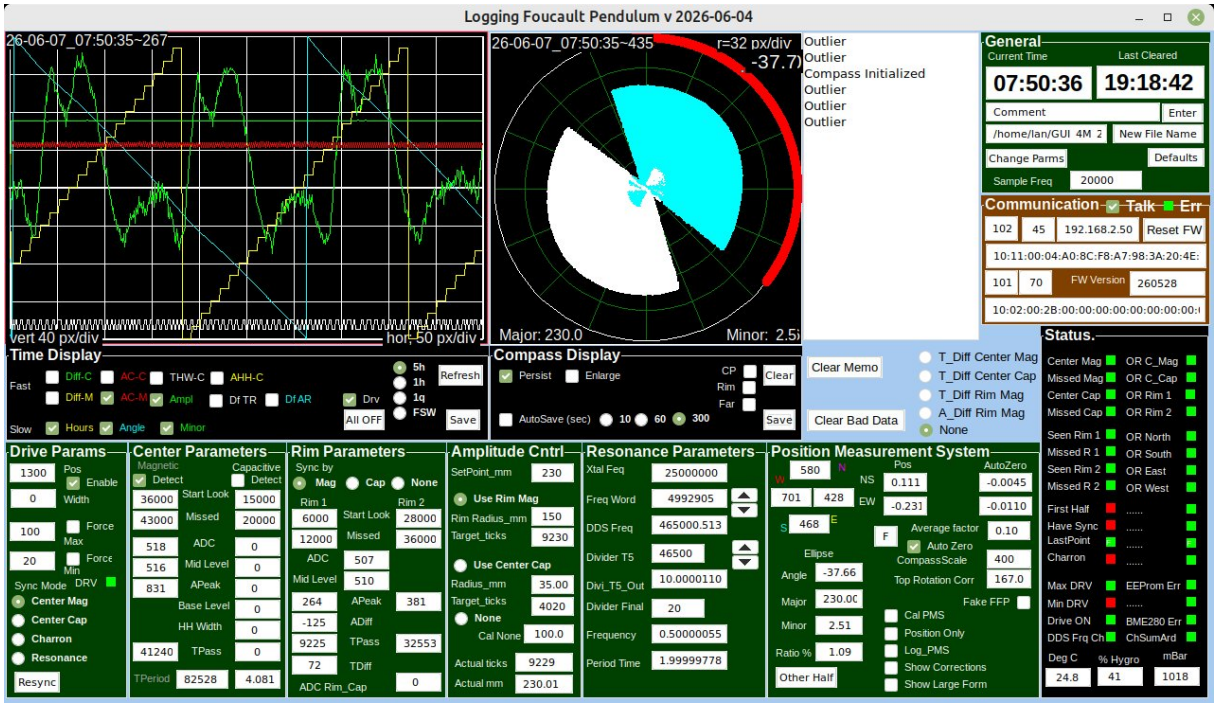

Fig 1. Overview of the GUI_Pendulum, the “Dashboard”.

Yes, at first sight it looks complex, but in fact it is much better structured than in the older version. Besides that, it supports many more functions and options.

From left to right, top to bottom:

1/ Time display, with Settings Panel.
Gives a display as function of time of several parameters.
The data enter the graph on the right side and are shifted left 1 pixel at each new entry.
The display area is 500 x 320 pixels, 50 (X) and 40 (Y) pixels per div.
Certain parameters may need better scaling. This can be done by editing the unit u_timedisplay.
There are two time scales:
Fast: one entry per halfswing of the pendulum.
Slow: Selectable: 1 quarter, 1 hour or 5 hour per division. FSW is for diagnostic purposes and gives 1 pixel per Full Swing.
The data are not averaged over the longer periods, just a sample is taken.

First row: Capacitive detection

Diff-C: Time Difference between successive Centerpasses Capacitive.
Gives info about the (ex)centricity of the center electrode.

AC_C: Peak Amplitude of CenterPass Capacitive signal.
This tells us a.o. about the height of the bob and the elipticity.

THW-C: Time of Half Height Width of Centerpasses Capacitive.
This tells us a.o. about the velocity of the bob and the elipticity.

AHH_C: Level of Half Height of the CenterPass Capacitive pulse.
The HH level is derived from the peak level and the base level.

Drv: Drive strength as Maximal / Minimal level.
If the Min / Max ratio differs much from 50% the Drive-Strength parameters should be reconsidered.

Second row: Magnetic detection

Diff_M: Time Difference between successive Centerpasses Magnetic.

AC_M: Peak Amplitude of Centerpass Magnetic signal.
This tells us a.o. about the height and the velocity of the bob and the elipticity.

Ampl: Pendulum Amplitude derived from RimPasses Magnetic.

Df_TR: Time Difference between succesive Rim1-passes.

Df_AR: Peak Amplitude difference between succesive Rim1-passes.

Slow:

Hours: Time of the day in 1-hour steps (Staircase)

Angle: Precession Angle.
Top: 180 West, +90 North, Center: 0 East, -90 South, Bottom: -180 West.

Minor: Minor axis of ellipse, incl. rotational direction of the ellipse.
Positive = CCW, negative = CW.

In the code (u_timedisplay.pas) are instructions to add / change parameters and scaling for the display.

The button **Refresh** refreshes the display manually. (normally this is done at each halfswing)
The button **Save** causes a .png image of the display to be saved on the harddisk of the GUI PC. The saved images will have a DateTimeStamp as the filename.

2/ Compass display, with Settings Panel
Gives a live display of the bob's movement in the horizontal plane.
If the PMS is adjusted correctly the top represents North, etc.
The circles have radii of 4, 32, 64, 96, 128 and 160 pixels.
On each message arrival (10 Hz) the position of the bob is plotted on the display.
The dots either persist, or disappear when the next message arrives. (Checkbox Persist)
During *HalfSwing == true* the dots are light blue, otherwise white.

The checkboxes on the right allow certain diagnostic info to be displayed with a red dot.

CP: A dot indicating the position measured at the CenterPass.

Rim: A dot at the positions of the Rim passes.

Far: A dot at the blue peak indicates the data point the farest away from the center.

Note that these position indicators are a bit off due to the latency of the message mechanism.

A small red circle on the outer circle of the display indicates the direction of the swinging plane = Precession Angle.

The checkbox **Enlarge** increases the scale of the display by 4. This is helpfull while adjusting the gains in the PMS.

The checkbox **AutoSave** enables the display to be saved as a .png image each 10, 60 or 300 seconds. When autosave is triggered the display is first cleared, then filled with bob positions for a few full swings, and then saved. The animated images on the website are made using this feature.

The **Save** button causes the image to be saved as is, without first clearing the display. The saved images will have a DateTimeStamp as the filename.

3/ Memo.
Shows certain alpha-numerical messages. Some controls are below the memo field.
The button **ClearMemo** clears the Memo field.

The **RadioButtons** allow values of the succesive Centerpass times to be shown in the memo field. Either from the Capacitive or Magnetic Centerpass detection or the differences in signal amplitude from the succesive Rim1-passes.

T_Diff_Center Mag: we see from the magnetic Centerpass detector:
Time of CenterPass, *Difference* with the *previous pass*, *PeriodTime*.
This tells us about the centricity of the Center detection coil.

T_Diff_Center Cap: we see from the Capacitive Centerpass detector:
Time of CenterPass, *Difference* with the *previous pass*, *PeriodTime*.
This tells us about the centricity of the Center detection electrode.

T_Diff_Rim_Mag: we see from the magnetic RimPass detector:
Time of Rim1Pass in the first Halfswing, second Halfswing, *Difference*
This tells us about the centricity of the Rim Coil.

A_Diff_Rim Mag: we see from the magnetic RimPass detector:
Peak Amplitude *of Rim1Pass* in the first Halfswing, second Halfswing, *Difference*
This tells us about the horizontality of the Rim Coil.

The differences in CenterPass and RimPass times can be used to check / adjust the centering of the detection coil or electrode. Investigate the differences with the pendulum launched in two perpendicular directions.

The differences in the peak signal amplitude from the Rim Coil in succesive passes tell us about the horizontality of the RimCoil, and likely also the other coils. We'll see the value of the most recent pass and the difference with the previous pass.
Investigate the differences with the pendulum launched in two perpendicular directions.

4/ General.
Shows Time-of-the-day, Last time the Compass Display was cleared,

Allows entry of a comment into the Logfile:
Write a comment and press the adjacent **Enter** button. (not the one on the keyboard!)
The comment is entered in both the data logfile and the events logfile.
New File Name Allows to start new logfiles by generating a new filename.
LogfileNames are automatically generated at the start of the program and at midnight, and have a DateTimestamp as filename.

Button **Change Parameters** causes the current settings to be transferred to the Arduino. There most of them will also be stored in EEPROM to be used after a reboot. The parameters are also written to the Parameter file, to be read at the startup of the program, and in an entry into the EventLogfile.

Button **Defaults** sets pre-programmed default values for many parameters. You can set your own defaults by editing the source code in unit u_main.pas
The field Sample Freq shows the sample frequency used by the GUI for several calculations. You cannot change it here. If it needs changing you should do that in the Arduino Firmware and in the GUI Defaults settings.

5/ Communication.
The Checkbox **Talk** starts / stops the Ethernet communication with the Arduino on the BobControl board. Starting / stopping is logged in the EventLog file.

The button **Reset FW** causes the Arduino on the BobControl Board and other hardware there to be reset.

Other fields show the number and length of the messages sent to the Arduino, the IP address of the Arduino, the first few byte values in the message sent to the Arduino, the number and length of the message received from the Arduino, the version of the Arduino Firmware and the first byte values in the message received from the Arduino.
The byte values are in Hexadecimal notation.
The Error indicator will be red when there is a CheckSum error in the incoming message.

6/ Drive Parameters.
All numeric values are in sample ticks.

Drive Position: #ticks from CenterPass to Mid of Drive Pulse.

Drive Width: #ticks for the width of the Drive Pulses.

Checkbox **Enable:** Enable / disable the Drive Pulses.

Max: PWM value for the maximal current of the Drive Pulse to be used by the amplitude control mechanism. Range 0..1023 produces 0 to 5 Volts as control voltage for the voltage to current converter.

Min: Minimum PWM value for the same purpose.

The checkboxes **Force** set the Min or Max values, overruling the Amplitude Control Mechanism.

The radiobuttons let us chose between sources of Drive Timing,
CenterPasses Magnetic,
CenterPasses Capacitive,
Charron Ring, (not yet implemented)
Resonance Mode. (not yet implemented)

The **Resync** button sends a signal to the Arduino to do a complete startover of the synchronisation process.

The **DRV indicator** flashes when a DrivePulse is given.

7/ Center Parameters.
All numeric values are in sample ticks.

The checkboxes **Detect** enable/disable the detectors for CenterPasses_Magnetic or CenterPasses_Capacitive. The detectors can work simultaneously.

Separate for Magnetic or Capacitive CenterPass detection we can set values for:

StartLook: Start looking for a Pass from this value of the PositionCounter. This is to prevent false triggers on the induction of the Drive Coil.

Missing: When the PositionCounter reaches this value we decide that the CenterPass has been missed. This results in a retry as if the button Resync was pressed.

ADC: the actual value of the CenterPass signal.

MidLevel:
For the Capacitive method the HalfHeight level at which the width of the pulse is measured. Calculated from averaged base- and Peak levels.
For the Magnetic method the level at which the CenterPass is detected. Measured at the end of the halfswings, when the bob is far away.

APeak: The peak level of the most recent CenterPulse.

Base Level: For the Capacitive method only: the signal level when the bob is far away. It is used to calculate the HalfHeight level for the HH pulsewidth measurement.

HH Width: Half Height Width of the Capacitive Center Pulse. This is a measure for the velocity of the bob, and so for the amplitude of the pendulum.

TPass: The HalfSwing time of the most recent Centerpass.

TPeriod: The period time (full swing) derived from either the magnetic or capacitive CenterPass detector, depending on which is selected to synchronise the DriveTiming.

8/ Rim Parameters.
All values are in sample ticks.
Only Magnetic Rim Pass detection is implemented. Capacitive Rim Pass detection is only prepared in the hardware. The signal is available in the software but nothing is done with it.
On top are the RadioButtons to select which CenterPass Detector will synchronize the RimPass detector:
Mag: The CenterPass Magnetic will synchronise the RimPass detection.
Cap: The CenterPass Capacitive will synchronise the RimPass detection.
None: The RimPass detector is disabled.

Separate for first and second RimPass:

StartLook: Start looking for a RimPass from this value of the PositionCounter. If we start looking to early the induction from the Drive Coil may cause a false reading.

Missed: If the PositionCounter reaches this value we decide that the RimPass is missed.

ADC: The actual value of the signal from the Rim Coil.

MidLevel: The RimCoil signal when the bob is at the end of the swing. This level is used to calculate the absolute peak amplitudes of the Rimcoil signal. The signal is averaged over many periods of the pendulum.

APeak: The absolute peak values of the rimcoil signal at the passes.

ADiff: The difference between the APeak of the last and the previous Rim1-Pass. This difference tells us something about the horizontality of the RimCoil.

TPass: The value of the PositionCounter at the Passes.

TDiff: The difference between the last and the previous Rim1 Pass times. This can tell us about the centering of the RimCoil.

ADC_Rim_Cap: The actual signal from the Capacitive Rim detection channel. No more than this is implemented for Capacitive Rim Pass detection.

9/ Amplitude Control.
With the radiobuttons the method of automatic amplitude control can be selected to use either the magnetic **RimPass** time, the width of the **Capicitive Center** pulse, or **no** automatic control.

SetPoint_mm: the target value for the pendulum's amplitude in mm.

For Amplitude control based on Magnetic Rimcoil Passes:
Rim Radius_mm: The radius of the RimCoil used.

Target ticks: The calculated value in ticks for TPassRim1. This value is calculated from SetPoint, RimcoilRadius, sample frequency and the measured period time of the pendulum.
If TPassRim1 > TargetTicks the amplitude is to small, and Max Drive current is used.
If TPassRim1 < TargetTicks the amplitude is to large, and Min Drive current is used.

For Amplitude control based on Capacitive CenterPulse width:
Note that this option is experimental and not yet completely implemented and tested.
Radius_mm: The radius of the Center electrode. The radius of the bob's electrode should be the same.

Target_ticks: The calculated target value for the HalfHeight Width of the Center Pulse.
If HHWidth > TargetTicks the the amplitude of the pendulum is to small and Max Drive current is used.
If HHWidth < TargetTicks the the amplitude of the pendulum is to large and Min Drive current is used.
Note: Currently the calculation of Target_ticks from the electrode radius is no more than an educated guess because of a substantial influence of electrode edge effects. Adjust Electrode radius until the desired amplitude is reached. Observe the bob itself for this, the reverse calculation is based on the same educated guess.

Cal_mm provides a calibration factor for when no amplitude control is used. It converts the normalized values from PMS-units to mm.

Without Amplitude Control the amplitude of the pendulum will depend solely on the amount of energy delivered to the bob and the damping the bob experiences, mainly air friction.
The amount of energy delivered to the bob depends on the height of the bob above the DriveCoil, the drive strength (Force Min or Force Max should be active), the Drive Current setting, the Width of the Drive pulse, the timing (DrivePosition) w.r.t. the radius of the

DriveCoil and the position of the bob at the driving moment. The latter depends on the bob's amplitude, so the effect could either increase or decrease with the amplitude.
In other words, it depends on many factors whether there will be a stabilizing effect.

10/ Resonance Parameters.
Note that this option is experimental and not yet tested in a pendulum.
The implementation is still incomplete.

The idea is to drive the pendulum with the frequency of the major axis of the ellipse, and not excite the slightly higher frequency of the minor ellipse axis. This requires a very stable frequency source which can be adjusted in very small steps, and a pendulum with a rather high Q.
For this we have the Direct Digital Synthesizer (DDS) which generates the 465 kHz signal for the PMS. This frequency can be adjusted in steps of approximately 0.1 Hz by changing the 28 bit FrequencyWord. The 465 kHz signal is used as a clock signal for the Arduino Timer 5 which devides that frequency by an adjustable value. That signal goes to a software divider called DeviderFinal which triggers the Drive pulses.

Xtal FReq: the frequency of the crystal oscillator on the DDS module; 25 Mhz.

FrequencyWord: The 28 bit Frequency Word for the DDS.

DDS Freq: The frequency the DDS produces. This is the frequency of the signal on the pendulum's wire and used for the PMS. It should not differ more that a few 100 Hz from 465 kHz.

Divider T5: The division factor for Timer T5.

Div_T5_Out: The output frequency of divider T5.

Divider_Final: The division factor for Divider_Final.

Frequency and **Period Time:** for the final Drive Pulses to be generated.

The values **FrequencyWord** and **Devider_T5** can be adjusted with the up-down buttons.
Note that the value **Divider_T5** must be < 65535, the range of a 16 bit unsigned integer.

11/ Position Measurement System.
Upper left we see 4 fields which show the actual values of the 4 signals **North**, **South**, **West**, **East**. These are ADC values from the 10 bit A/D converter in the Arduino, so they range from 0 to 1023.

The **Pos** fields show the normalized values **PosNS** and **PosEW.**

The fields **AutoZero** show the correction values when AutoZero is enabled.
This corrects the PosNS and PosEW values for small DC offsets.
At this moment I cannot oversee what influence a DC offset will have on the accuracy of the whole system.

The field **Average factor** allows to set the average factor for the autozero corrrection.
The value must be 0 <= factor <= 1. The lower the factor the more averaging is done and the slower the correction reacts. A value of 0.1 will do in most cases.

Compass Scale sets a value for the scale of the compass display.

The fields under "Ellipse" show the values for
Precession Angle in degrees (counting CCW up from East = zero).

Length of the **Major** ellipse axis in mm, using the applicable calibration.

Length of the **Minor** ellipse axis in mm, using the applicable calibration.

Ratio Minor / Major axis or the ellipticity in %.

The Checkboxes do:
Auto Zero: Enable / Disable Auto Zero corrections. When disabled the calculation and logging of the correction values will continue.

Fake FFP: Fake a Fast Foucault Precession of 1 deg / sec for testing the algorithms. The fake is done by decreasing (for the Northern hemisphere) the value of TopRotationCorrection.

Cal PMS: For calibrating the gains of the 4 PMS receivers. This procedure should be done with the bob in the rest position. The display shows 4 differently colored blips, 1 for each receiver. Adjust the gains such that the blips stay in the center.
During this adjustment it can be convenient to show the **LargeForm** with the numerical values.

Position Only: Only Position data are processed. Combine this with **Cal Zero**.

Log PMS: Writes a separate logfile with only PMS data at a rate of 1 entry per incoming message.

Show Corrections: Show a form where correction data for the PMS can be entered. Background: The PMS suffers from a small amount of non linearity and cross talk. (pillow shaped distorsion) This is a preparation for a future correction algorithm. (no further implementation yet)

Show Large Form: Show a large form with the ADC values. This is helpfull while adjusting the gains of the receivers and so. The large form can be read from quite some distance.

12/ Status.
The Status pane shows the values of the bits in the 32 bit Status word.
green = 0, red = 1.

Seen Center and **Missed Center** for **Mag**netic and **Cap**acitive detection. Flash shortly.

Seen and **Missed Rim 1** and **Rim 2**. Flash shortly.

FirstHalf: We are in the “blue” HalfSwing.

HaveSync: The active Center Pass Detector is in sync with the bob.

LastPoint: flashes when the bob does the CenterPass into the FirstHalf.

Using **Max** or **Min** Drive Current as dictated by the Amplitude Control mechanism. (none when Drive is disabled). Also reacts on Force Max or Min).

DriveON: flashes when there is a DrivePulse given. This indicator is repeated in the Drive-Parm pane.

DDS Frq Ch: When the Resonance Tracking Mechanism (not yet implemented) has changed the DDS frequency.

Out-of-Range indication for the Center and Rim detection signals. (Flash shortly)
Out of range means that the signal reaches 0 or 1023, the limits of the A/D converter.

EEPRom-Err: An indication that there was a CheckSum error while reading the EEPROM in the Arduino.

BME280 Err: The climate sensor in the topunit could not be initialized.

CheckSumArd: The Arduino has detected a CheckSum error in the message from the GUI.

At the bottom the **temperature**, **relative humidity** and **barometric pressure** as measured by the BME 280 climate sensor in the topunit.

Note:
Most controls (textboxes, checkboxes, etc) have a so called hint, which appears when you put the mouse pointer above the control, but without clicking.
The hint gives some information about the purpose of the control.
For controls to enter parameters or settings the hint can start with one or two asterisks.
* This parameter is saved in the Parameter file and retreived at startup of the program.
** This parameter is also saved in EEPROM of the Arduino on the BobControl Board and retreived when the Arduino reboots.